bdiHCI

BDM interface for HI-WAVE™ Debugger

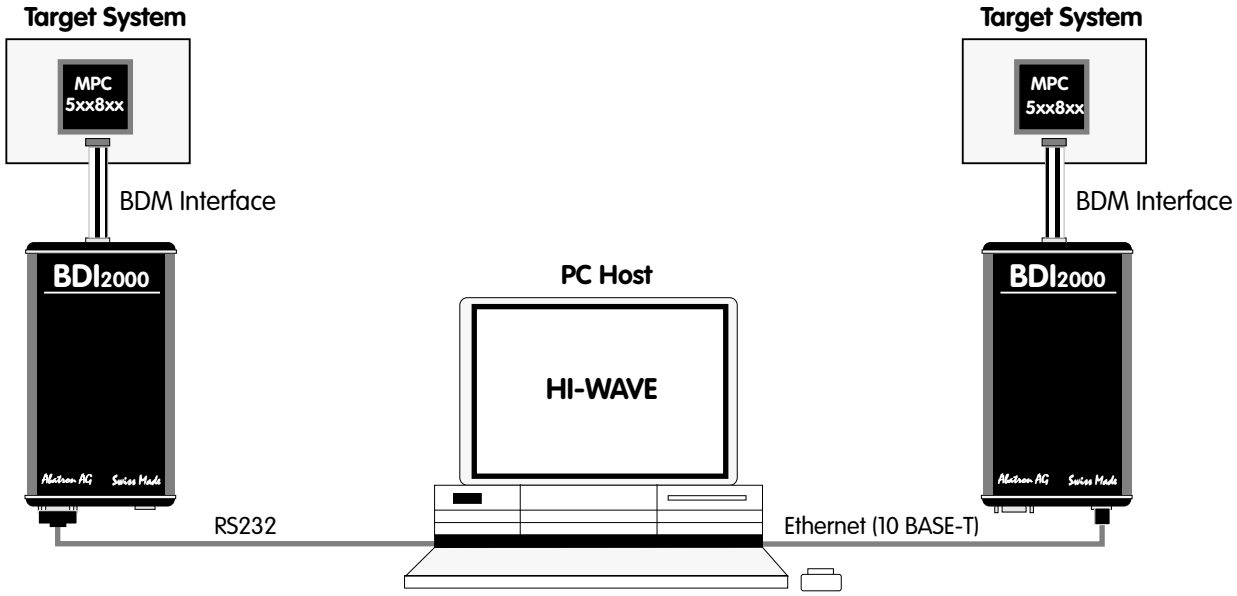
PowerPC MPC8xx/MPC5xx

User Manual

Manual Version 1.06 for BDI2000

Appendices

1 Introduction



The BDI2000 adds Background Debug Mode features to the HI-WAVE debugger environment. With the BDI2000, you control and monitor the microcontroller solely through the stable on-chip debugging services. You won't waste time and target resources with a software ROM monitor, and you eliminate the cabling problems typical of ICE's. This combination runs even when the target system crashes and allows developers to continue investigating the cause of the crash.
A RS232 interface with a maximum of 115 kBaud and a 10Base-T Ethernet interface is available for the host interface.
The bdiHCI setup software is used to update the firmware and to configure the BDI2000 so it works with the HI-WAVE debugger.

1.1 BDI2000

The BDI2000 is a processor system in a small box. It implements the interface between the BDM pins of the target CPU and a 10Base-T Ethernet / RS232 connector. BDI2000 is powered by a MC68360, 512Kbyte RAM and a flash memory of 1024Kbyte. As a result of consistent implementation of lasted technology, the BDI2000 is optimally prepared for further enhancements. The firmware and the programmable logic of the BDI2000 can be updated by the user with a simple Windows based configuration program. The BDI2000 supports 1.8 – 5.0 Volts target systems (3.0 – 5.0 Volts target systems with Rev. A/B).

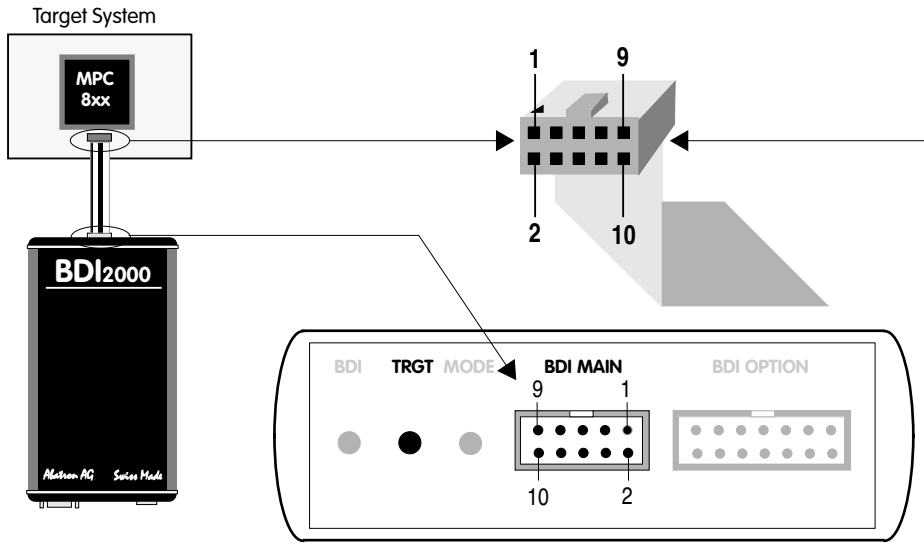
2 Installation

2.1 Connecting the BDI2000 to Target

The cable to the target system is a ten pin flat ribbon cable. In case where the target system has an appropriate connector, the cable can be directly connected. The pin assignment is in accordance with the Motorola specification.

In order to ensure reliable operation of the BDI (EMC, runtimes, etc.) the target cable length must not exceed 20 cm (8").

Rev. A *«Rev. A» is the first BDI2000 version, produced until June 1999*

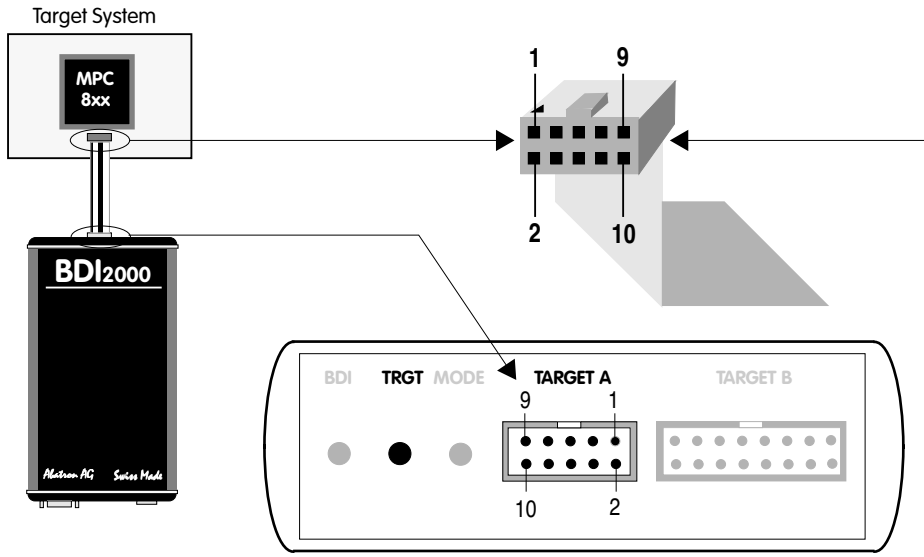


Target Connector

1 - VFLS0
2 - $\overline{\text{SRESET}}$
3 - GROUND
4 - DSCK
5 - GROUND
6 - VFLS1
7 - $\overline{\text{HRESET}}$
8 - DSDI
9 - Vcc Target
10 - DSDO

The green LED «TRGT» marked light up when target is powered up

Rev B/C



Target Connector

1 - VFLS0
2 - $\overline{\text{SRESET}}$
3 - GROUND
4 - DSCK
5 - GROUND
6 - VFLS1
7 - $\overline{\text{HRESET}}$
8 - DSDI
9 - Vcc Target
10 - DSDO

The green LED «TRGT» marked light up when target is powered up

For BDI MAIN / TARGET A connector signals see table on next page.

BDI MAIN / TARGET A Connector Signals:

Pin	Name	Describtion
1	VFLS0	These pin and pin 6 (VFLS1) indicate to the debug port controller whether or not the MPC is in debug mode. When both VFLS0 and VFLS1 are at "1", the MPC is in debug mode.
2	$\overline{\text{SRESET}}$	This is the Soft-Reset bidirectional signal of the MPC8xx. On the MPC5xx it is an output. The debug port configuration is sampled and determined on the rising-edge of $\overline{\text{SRESET}}$ (for both processor families). On the MPC8xx it is a bidirectional signal which may be driven externally to generate soft reset sequence. This signal is in fact redundant regarding the MPC8xx debug port controller since there is a soft-reset signal integrated within the debug port protocol. However, the local debug port controller uses this signal for compatibility with MPC5xx existing boards and s/w.
3+5	GND	**System Ground**
4	DSCK	**Debug-port Serial Clock** During asynchronous clock mode, the serial data is clocked into the MPC according to the DSCK clock. The DSCK serves also a role during soft-reset configuration.
6	VFLS1	These pin and pin 1 (VFLS0) indicate to the debug port controller whether or not the MPC is in debug mode. When both VFLS0 and VFLS1 are at "1", the MPC is in debug mode.
7	$\overline{\text{HRESET}}$	This is the Hard-Reset bidirectional signal of the MPC. When this signal is asserted (low) the MPC enters hard reset sequence which include hard reset configuration. This signal is made redundant with the MPC8xx debug port controller since there is a hard-reset command integrated within the debug port protocol.
8	DSDI	**Debug-port Serial Data In** Via the DSDI signal, the debug port controller sends its data to the MPC. The DSDI serves also a role during soft-reset configuration.
9	Vcc Target	**1.8 – 5.0V:** This is the target reference voltage. It indicates that the target has power and it is also used to create the logic-level reference for the input comparators. It also controls the output logic levels to the target. It is normally fed from Vdd I/O on the target board. **3.0 – 5.0V with Rev. A/B :** This input to the BDI2000 is used to detect if the target is powered up. If there is a current limiting resistor between this pin and the target Vdd, it should be 100 Ohm or less.
10	DSDO	**Debug-port Serial Data Out** DSDO is clocked out by the MPC according to the debug port clock, in parallel with the DSDI being clocked in. The DSDO serves also as "READY" signal for the debug port controller to indicate that the debug port is ready to receive controller's command (or data).

Mention of sources used: MPC860ADS User's Manual, Revision A

Enhanced Debug Mode Detection:

For MPC8xx and MPC555 targets, debug mode (Freeze) detection also works when the BDM connector pins VFLS0 and VFLS1 are not connected to the target. If not connected to VFLSx, this BDM connector pins should be left open or tied to Vcc. The BDI uses the following algorithm to check if the target is in debug mode (freezed):

```
BOOL PPC_TargetFreezed(void) {
  if ((VFLS0 != 1) | (VFLS0 != 1)) return FALSE;
  read debug port status;
  if (status == freezed) return TRUE;
  else                   return FALSE;
```


2.1.1 Changing Target Processor Type

Before you can use the BDI2000 with an other target processor type (e.g. CPU32 <--> PPC), a new setup has to be done (see Appendix A). During this process the target cable must be disconnected from the target system. The BDI2000 needs to be supplied with 5 Volts via the BDI OPTION connector (Rev. A) or via the POWER connector (Rev. B/C). For more information see chapter 2.2.1 «External Power Supply».

To avoid data line conflicts, the BDI2000 must be disconnected from the target system while programming the logic for an other target CPU.

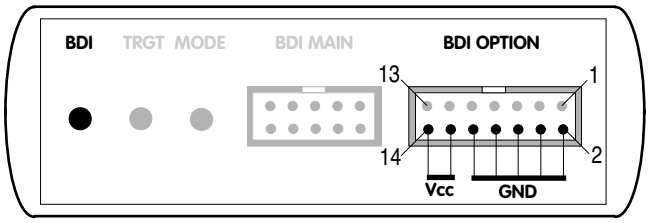
2.2 Connecting the BDI2000 to Power Supply

2.2.1 External Power Supply

The BDI2000 needs to be supplied with 5 Volts (max. 1A) via the BDI OPTION connector (Rev. A) or via POWER connector (Rev. B/C). The available power supply from Abatron (option) or the enclosed power cable can be directly connected. In order to ensure reliable operation of the BDI2000, keep the power supply cable as short as possible.

For error-free operation, the power supply to the BDI2000 must be between 4.75V and 5.25V DC. **The maximal tolerable supply voltage is 5.25 VDC. Any higher voltage or a wrong polarity might destroy the electronics.**

Rev. A

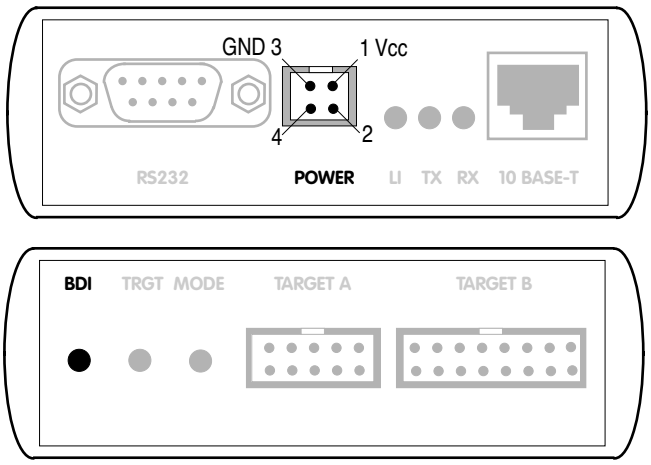


The green LED «BDI» marked light up when 5V power is connected to the BDI2000

BDI OPTION Connector

1 - NOT USED
2 - GROUND
3 - NOT USED
4 - GROUND
5 - NOT USED
6 - GROUND
7 - NOT USED
8 - GROUND
9 - NOT USED
10 - GROUND
11 - NOT USED
12 - Vcc (+5V)
13 - Vcc Target (+5V)
14 - Vcc (+5V)

Rev. B/C



The green LED «BDI» marked light up when 5V power is connected to the BDI2000

POWER Connector

1 - Vcc (+5V)
2 - VccTGT
3 - GROUND
4 - NOT USED

Please switch on the system in the following sequence:

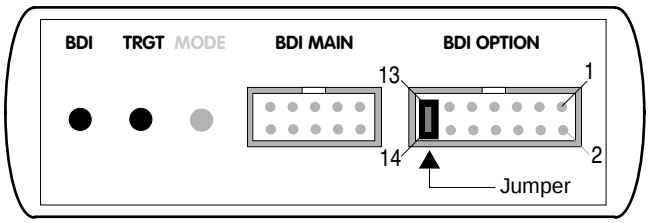
- 1 --> external power supply
- 2 --> target system

2.2.2 Power Supply from Target System

The BDI2000 needs to be supplied with 5 Volts (max. 1A) via BDI MAIN target connector (Rev. A) or via TARGET A connector (Rev. B/C). This mode can only be used when the target system runs with 5V and the pin «Vcc Target» is able to deliver a current up to 1A@5V. For pin description and layout see chapter 2.1 «Connecting the BDI2000 to Target». Insert the enclosed Jumper as shown in figure below. **Please ensure that the jumper is inserted correctly**.

For error-free operation, the power supply to the BDI2000 must be between 4.75V and 5.25V DC. **The maximal tolerable supply voltage is 5.25 VDC. Any higher voltage or a wrong polarity might destroy the electronics.**

Rev. A

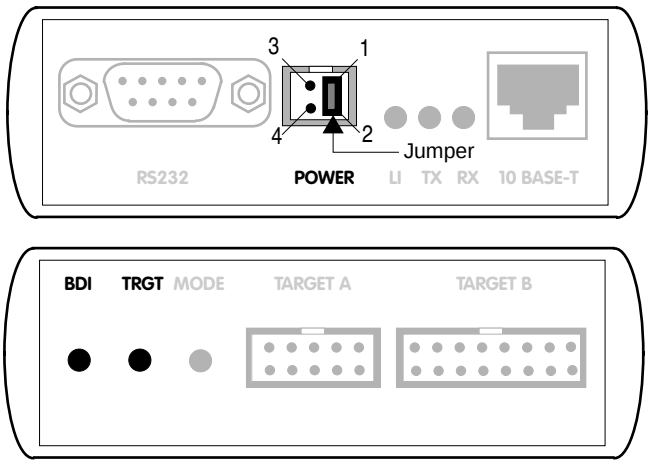


The green LEDs «BDI» and «TRGT» marked light up when target is powered up and the jumper is inserted correctly

BDI OPTION Connector

1 - NOT USED
2 - GROUND
3 - NOT USED
4 - GROUND
5 - NOT USED
6 - GROUND
7 - NOT USED
8 - GROUND
9 - NOT USED
10 - GROUND
11 - NOT USED
12 - Vcc (+5V)
13 - Vcc Target (+5V)
14 - Vcc BDI2000 (+5V)

Rev. B/C



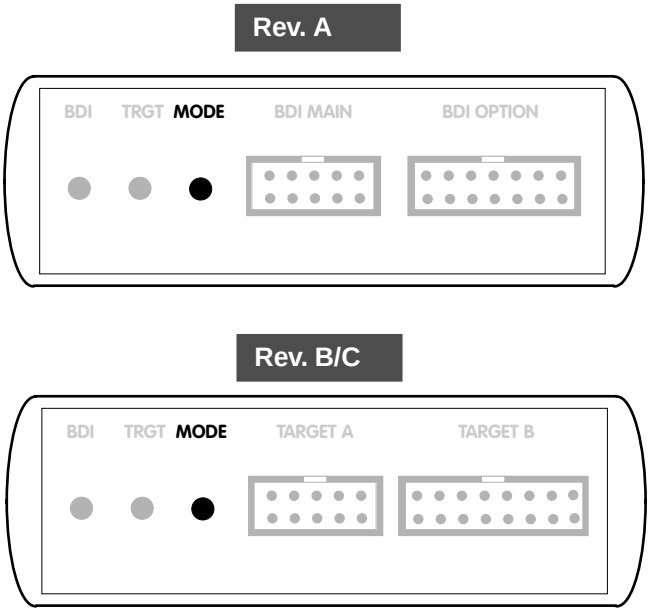
The green LEDs «BDI» and «TRGT» marked light up when target is powered up and the jumper is inserted correctly

POWER Connector

1 - Vcc BDI2000 (+5V)
2 - Vcc Target (+5V)
3 - GROUND
4 - NOT USED

2.3 Status LED «MODE»

The built in LED indicates the following BDI states:



MODE LED	BDI STATES
OFF	The BDI is ready for use, the firmware is already loaded.
ON	The power supply for the BDI2000 is < 4.75VDC.
BLINK	The BDI «loader mode» is active (an invalid firmware is loaded or loading firmware is active).

2.4 Connecting the BDI2000 to the Host

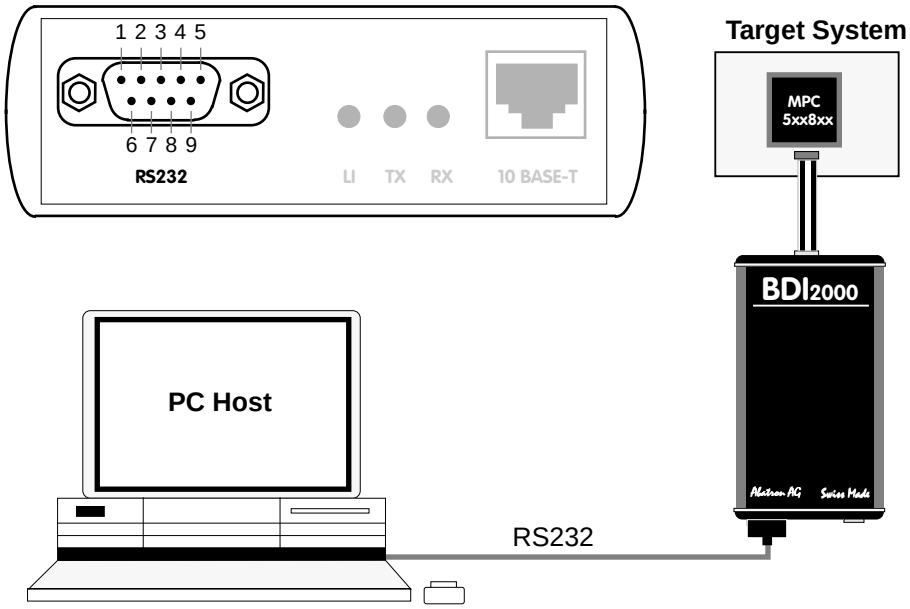
2.4.1 Serial line communication

The host is connected to the BDI through the serial interface (COM1...COM4). The communication cable between BDI and Host is a serial cable (RXD / TXD are crossed). There is the same connector pinout for the BDI and for the Host side (Refer to Figure below).

Rev. A

RS232 Connector
(for PC host)

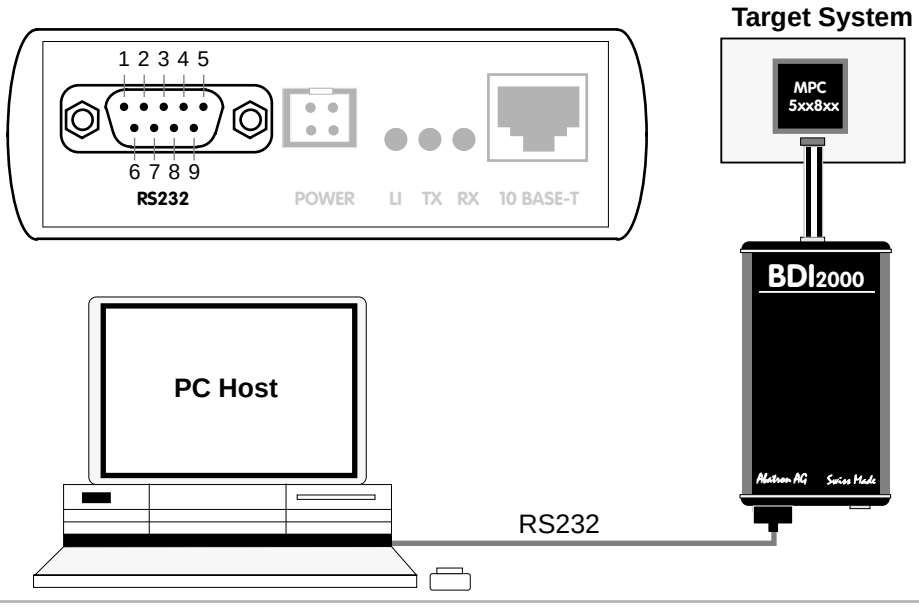
1 - NC
2 - RXD data from host
3 - TXD data to host
4 - NC
5 - GROUND
6 - NC
7 - NC
8 - NC
9 - NC



Rev. B/C

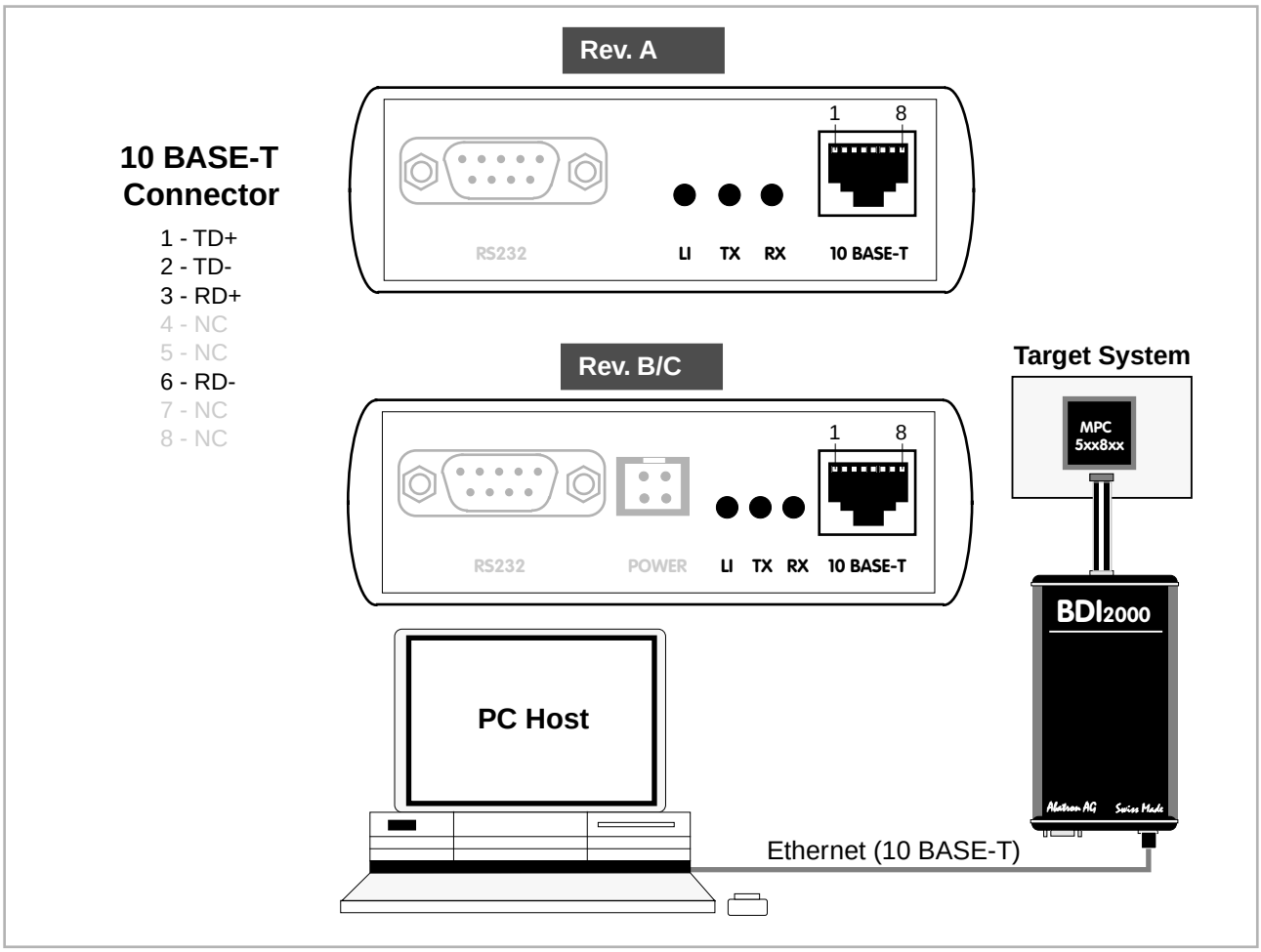
RS232 Connector
(for PC host)

1 - NC
2 - RXD data from host
3 - TXD data to host
4 - NC
5 - GROUND
6 - NC
7 - NC
8 - NC
9 - NC



2.4.2 Ethernet communication

The BDI2000 has a built-in 10 BASE-T Ethernet interface (see figure below). Connect an UTP (Unshilded Twisted Pair) cable to the BD2000. For thin Ethernet coaxial networks you can connect a commercially available media converter (BNC-->10 BASE-T) between your network and the BDI2000. Contact your network administrator if you have questions about the network.



The following explains the meanings of the built-in LED lights:

LED	Name	Description
LI	Link	When this LED light is ON, data link is successful between the UTP port of the BDI2000 and the hub to which it is connected.
TX	Transmit	When this LED light BLINKS, data is being transmitted through the UTP port of the BDI2000
RX	Receive	When this LED light BLINKS, data is being received through the UTP port of the BDI2000

2.5 Installation of the Configuration Software

On the enclosed diskette you will find the BDI configuration software and the firmware required for the BDI. Copy all these files to a directory on your hard disk.

The following files are on the diskette:

b20mpc.exe	Configuration program
b20mpc.hlp	Helpfile for the configuration program
b20ppcfw.xxx	Firmware for BDI2000 for MPC8xx/MPC5xx targets
ppcjed20.xxx	JEDEC file for the BDI2000 (Rev. A/B) logic device programming
ppcjed21.xxx	JEDEC file for the BDI2000 (Rev. C) logic device programming
bdiifc32.dll	BDI Interface DLL (32bit version)
*.bdi	Configuration Examples

Example of an installation process:

- Copy the entire contents of the enclosed diskette into a directory on the hard disk.
- You may create a new shortcut to the b20mpc.exe configuration program.

2.6 Configuration

Before you can use the BDI together with the debugger, the BDI must be configured. Use the *SETUP* menu and follow the steps listed below:

- Load or update the firmware / logic, store IP address --> *Firmware*
- Set the communication parameters between Host and BDI --> *Communication*
- Setup an initialization list for the target processor --> *Initlist*
- Select the working mode --> *Mode*
- Transmit the configuration to the BDI --> *Mode Transmit*

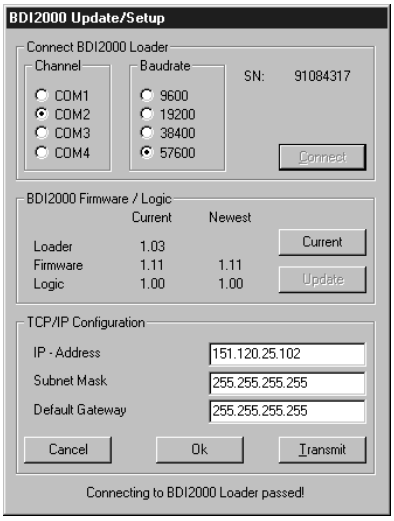
For information about the dialogs and menus use the help system (F1).

2.6.1 BDI2000 Setup/Update

First make sure that the BDI is properly connected (see Chapter 2.1 to 2.4). The BDI must be connected via RS232 to the Windows host.

To avoid data line conflicts, the BDI2000 must be disconnected from the target system while programming the logic for an other target CPU (see Chapter 2.1.1).

The following dialogbox is used to check or update the BDI firmware and logic and to set the network parameters.



dialog box «BDI2000 Update/Setup»

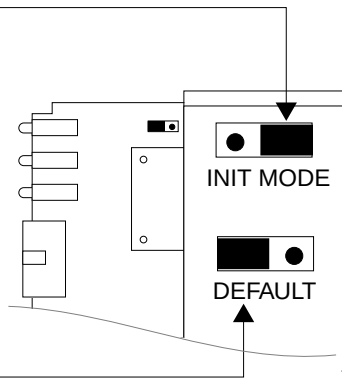
The following options allow you to check or update the BDI firmware and logic and to set the network parameters:

Channel	Select the communication port where the BDI2000 is connected during this setup session.
Baudrate	Select the baudrate used to communicate with the BDI2000 loader during this setup session.

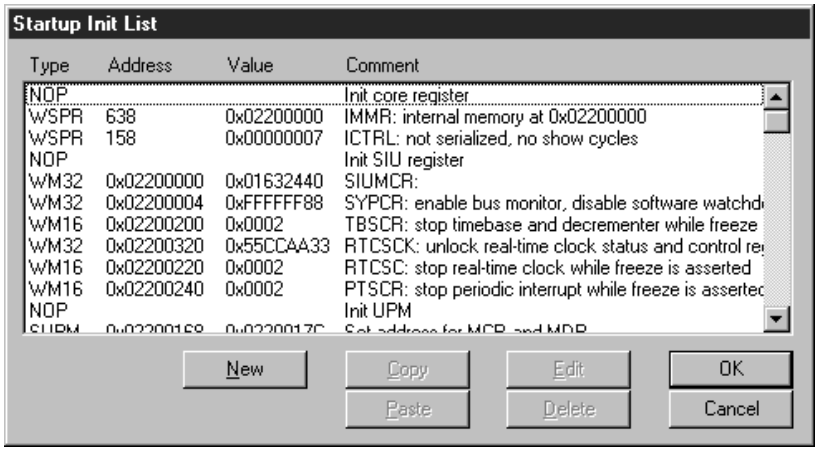
Connect	Click on this button to establish a connection with the BDI2000 loader. Once connected, the BDI2000 remains in loader mode until it is restarted or this dialog box is closed.
Current	Press this button to read back the current loaded BDI2000 software and logic versions. The current loader, firmware and logic version will be displayed.
Update	This button is only active if there is a newer firmware or logic version present in the execution directory of the BDI setup software. Press this button to write the new firmware and/or logic into the BDI2000 flash memory / programmable logic.
IP Address	Enter the IP address for the BDI2000. Use the following format: xxx.xxx.xxx.xxxe.g.151.120.25.101 Ask your network administrator for assigning an IP address to this BDI2000. Every BDI2000 in your network needs a different IP address.
Subnet Mask	Enter the subnet mask of the network where the BDI is connected to. Use the following format: xxx.xxx.xxx.xxxe.g.255.255.255.0 A subnet mask of 255.255.255.255 disables the gateway feature. Ask your network administrator for the correct subnet mask.
Default Gateway	Enter the IP address of the default gateway. Ask your network administrator for the correct gateway IP address. If the gateway feature is disabled, you may enter 255.255.255.255 or any other value..
Transmit	Click on this button to store the network configuration in the BDI2000 flash memory.

In rare instances you may not be able to load the firmware in spite of a correctly connected BDI (error of the previous firmware in the flash memory). **Before carrying out the following procedure, check the possibilities in Appendix «Troubleshooting»**. In case you do not have any success with the tips there, do the following:

- Switch OFF the power supply for the BDI and open the unit as described in Appendix «Maintenance»
- Place the jumper in the **«INIT MODE»** position
- Connect the power cable or target cable if the BDI is powered from target system
- Switch ON the power supply for the BDI again and wait until the LED «MODE» blinks fast
- Turn the power supply OFF again
- Return the jumper to the **«DEFAULT»** position
- Reassemble the unit as described in Appendix «Maintenance»



3 Init List



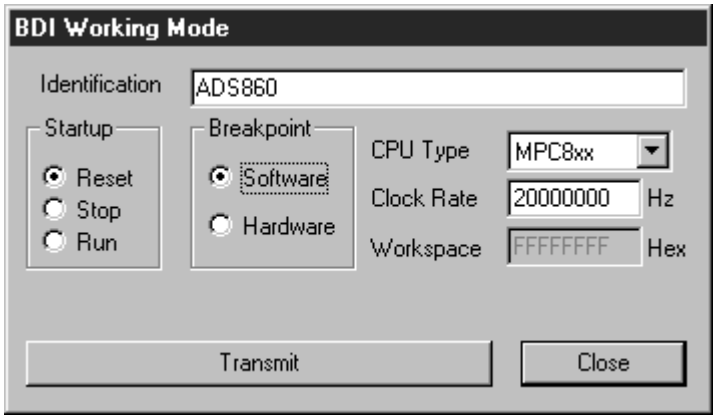
dialog box «Startup Init List»

In order to prepare the target for debugging, you can define an Initialization List. This list is stored in the Flash memory of the BDI2000 and worked through every time the target comes out of reset. Use it to get the target operational after a reset. The memory system is usually initialized through this list. After processing the init list, the RAM used to download the application must be accessible.

Use on-line help (F1) and the supplied configuration examples on the distribution disk to get more information about the init list.

Note:
You may also use the debuggers feature to setup the hardware. But keep in mind, that the BDI will speed up BDM communication clock after processing its own initialization list based on the «Clock Rate» field in the «BDI Working Mode» dialog box (see next chapter). The «Clock Rate» value you have to enter is therefore the value the target runs immediately after reset.
To use a BDM speed as fast as possible, you should at least speed up the target with an entry in the BDI initialization list and set the «Clock Rate» field to the appropriate value. The rest of the initialization (e.g. memory controller) can be done with the debugger.

4 BDI working modes



dialog box «BDI Working Mode»

With this dialog box you can define how the BDI interacts with the target system.

Identification	Enter a text to identify this setup. This text can be read by the debugger with the appropriate Command.
Startup	Startup mode defines how the BDI interacts with the target processor after reset or power up. The options RESET, STOP or RUN can be selected.
Breakpoint	Breakpoint mode defines how instruction breakpoints are implemented. When Software is selected (default), instruction breakpoints are set as requested by the debugger (Software or Hardware breakpoints). When Hardware is selected, the BDI uses always hardware breakpoints. This is useful when the attached debugger does not support hardware breakpoints on instruction access.
CPU Type	Select the CPU type of the target system.
Clock Rate	Enter the clock rate the target CPU runs after BDI has worked through the init list. BDI selects the BDM communication speed based on this parameter. If this parameter selects a CPU clock rate that is higher than the real clock rate, BDM communication may fail. When selecting a clock rate slower than possible, BDM communication still works but not as fast as possible.
Workspace	In order to access the floating-point registers of a MPC5xx microprocessor, the BDI needs a workspace of 8 bytes in target RAM. Enter the base address of this RAM area. This memory is used, when a floating-point register is accessed. If there is no RAM space available for the BDI, you may enter 0xFFFFFFFF as the workspace address but then, accessing floating-point registers is not possible.
Transmit	Click on this button to send the initialization list and the working mode to the BDI. This is normally the last step done before the BDI can be used with the debugging system.

4.1 Startup Mode

Startup mode defines how the BDI interacts with the target system after a reset or power up sequence.

4.1.1 Startup mode RESET

In this mode no ROM is required on the target system. The necessary initialization is done by the BDI with the programmed init list. The following steps are executed by the BDI after system reset or system power up:

- HRESET is activated on the target system.
- HRESET is deactivated and the target is forced into debug mode.
- The BDI works through the initialization list and writes to the corresponding addresses.

The RESET mode is the standard working mode. Other modes are used in special cases (i.e. applications in ROM, special requirements on the reset sequence...).

4.1.2 Startup Mode STOP

In this mode the initialization code is in a ROM on the target system. The code in this ROM handles base initialization. At the end of the code, the initialization program enters an endless loop until it is interrupted by the BDI. This mode is intended for special requirements on the reset sequence (e.g. loading a RAM based programmable logic device).
In this mode the following steps are executed by the BDI after system reset or power up:

- HRESET is activated on the target system.
- HRESET is deactivated and the target is forced into debug mode.
- The target is started and begins executing application code.
- After a delay of 2 seconds, the target is forced into debug mode.
- The BDI works through the initialization list and writes the corresponding addresses.

4.1.3 Startup mode RUN

This mode is used to debug applications which are already stored in ROM. The application is started normally and is stopped when the debugger is started.
In this mode, the following steps are executed by the BDI after system reset or power up:

- HRESET is activated on the target system.
- HRESET is deactivated and the target is forced into debug mode.
- The target is startet and begins executing application code.
- The application runs until it is stopped by the debugger.

5 Working with HI-WAVE

5.1 Setup

There are three environment variables concerning the interface between HI-WAVE and the BDI2000. The first one (BDI_LINK) defines the communication channel used. The second one (BDI_COMPRESS) defines if data compression should be used when downloading via an asynchronous communication channel (e.g. COM1). The third one (BDI_VERIFY) defines data verification during program download.

BDI_LINK

For communication via a serial communication port use the following initialization string:

Syntax: BDI_LINK=COMn baudrate

n:Number of used COM port (1,2,3,4)

baudrate:The baudrate (9600, 19200, 38400, 57600, 115200). Use the baudrate selected in menu *Setup->Communication*.

Example: BDI_LINK=COM1 57600

For communication via an Ethernet use the following initialization string:

Syntax: BDI_LINK=NETWORK ipaddress 1

ipaddress:The IP address of the BDI2000 (Format: xxx.xxx.xxx.xxx).

Example: BDI_LINK=NETWORK 151.120.25.101 1

BDI_COMPRESS

By default, data compression is enabled for asynchronous communication channels. With older computers, it's possible that download speed is faster without data compression. With this environment variable you can disable data compression.

Syntax: BDI_COMPRESS=ON | OFF | 0 | 1

Example: BDI_COMPRESS=OFF

BDI_VERIFY

This variable defines if download data is read back from target memory and compared against the written data.

Syntax: BDI_VERIFY=NO | FIRST | FULL | 0 | 1 | 2

NO (0):No read back at all.

FIRST (1):The first byte of every download block is read back and verified.

FULL (2):Every byte is read back and verified.

Example: BDI_VERIFY=FIRST

Example:

```
.....
BDI_LINK=NETWORK 151.120.25.101 1
BDI_COMPRESS=ON
BDI_VERIFY=FIRST
.....
```


5.2 Direct Commands

For special functions the BDI supports so called «Direct Commands». This commands can be used interactively in the HI-WAVE Command Line Window or entered in a command file (e.g. PRELOAD.CMD). This Direct Commands are not interpreted by HI-WAVE but directly sent to the BDI. After processing the command the result is displayed in the Command Line Window.

Use the following syntax to enter a Direct Command:

BDI <direct-command>

Direct Commands are ASCII - Strings with the following structure:

<Object>.<Action> [<ParName>=<ParValue>]...

Example:

flash.erase addr=0x02800000

All names are case insensitive. Parameter values are numbers or strings. Numeric parameters can be entered as decimal (e.g. 700) or as hexadecimal (0x80000) values.

5.2.1 Target.Reset

This direct command executes a real physical reset of the target system.

5.2.2 Flash.Setup

In order to support loading into flash memory, the BDI needs some information about the used flash devices. Before any other flash related command can be used, this direct command must be executed.

External flash memories:

Syntax:

flash.setup type=am29f size=0x80000 bus=32 workspace=0x1000

type	This parameter defines the type of flash used. It is used to select the correct programming algorithm. The following flash types are supported: AM29F, AM29BX8, AM29BX16, I28BX8, I28BX16, AT49, AT49X8, AT49X16, STRATAX8, STRATAX16, MIRROR, MORRORX8, MIRRORX16, I28BX32, AM29DX16, AM29DX32
size	The size of **one** flash chip in bytes (e.g. AM29F010 = 0x20000). This value is used to calculate the starting address of the current flash memory bank.
bus	The width of the memory bus that leads to the flash chips. Do not enter the width of the flash chip itself. The parameter TYPE carries the information about the number of data lines connected to one flash chip. For example, enter 16 if you are using two AM29F010 to build a 16bit flash memory bank.
workspace	If a workspace is defined, the BDI uses a faster programming algorithm that run out of RAM on the target system. Otherwise, the algorithm is processed within the BDI. The workspace is used for a 1kByte data buffer and to store the algorithm code. There must be at least 2kBytes of RAM available for this purpose.

MPC555 internal flash:

Syntax:

flash.setup type=mpc555 clock=20 workspace=0x007FC000

type	This parameter defines the type of flash used. Enter MPC555 or MPC555SHD when programming the MPC555 internal flash. MPC555SHD selects the shadow information.
clock	Enter the CPU clock frequency in MHz. The BDI needs this parameter to calculate the program / erase pulse timing. It is important to enter the correct frequency otherwise the erase / programming pulses are not correct.
workspace	This value defines the base address of a workspace in target RAM. The workspace is used for a data buffer and to store the algorithm code. There must be at least 2kBytes of RAM available for this purpose.

```
;Setup for 20MHz, use internal SRAM array B for workspace
flash.setup type=mpc555 clock=20 workspace=0x007FC000
```

MPC565 internal flash:

Syntax:

flash.setup type=uc3f workspace=0x007F8000

type	This parameter defines the type of flash used. Enter UC3F or UC3FSHD when programming the MPC565 internal flash. UC3FSHD selects the shadow information.
workspace	This value defines the base address of a workspace in target RAM. The workspace is used for a data buffer and to store the algorithm code. There must be at least 2kBytes of RAM available for this purpose.

```
;Setup for MPC565, use internal CALRAM A for workspace
flash.setup type=uc3f workspace=0x007F8000
```

5.2.3 Flash.Erase

External flash memories:
This command allows to erase one flash sector.

Syntax:

flash.erase addr=0x02800000 mode=chip

addr	The start address of the flash sector to erase.
mode	This parameter defines the erase mode. The following modes are supported: CHIP, BLOCK and SECTOR (default is sector erase)

MPC555 internal flash:
This command allows to erase one or multiple block(s) in one flash module.

Syntax:

flash.erase addr=0x004000FF

The BDI assumes the following structure of the address:

16 bit	1 bit	7 bit	8 bit
module address	C	<reserved>	block [0:7]

module address	The 16 most significant bits of the flash module address.
C	The censor bit. If this bit is set, the censor information is erased. **Use with caution!**
block	The bit mask to select the flash block to erase. Bit ordering is the same as in the CM-FCTL register (see MPC555 manual).

```
;Erase module A/B all sectors, flash memory is mapped to 0x00400000
flash.erase addr=0x004000FF
flash.erase addr=0x004400FC
```

MPC565 internal flash:
This command allows to erase one or multiple block(s) in one flash module.

Syntax:

flash.erase addr=0x004000FF

The BDI assumes the following structure of the address:

16 bit	1 bit	5 bit	2 bit	8 bit
module address	C	<reserved>	sbb[0:1]	block [0:7]

module address	The 16 most significant bits of the flash module address.
C	The censor bit. If this bit is set, the censor information is erased. **Use with caution!**
sbb*	The bit mask to select the small blocks to erase. Bit ordering is the same as in the UC3FCTL register (see MPC565 manual).
block	The bit mask to select the flash block to erase. Bit ordering is the same as in the UC3FCTL register (see MPC565 manual).

```
;Erase module A/B all sectors, flash mapped to 0x00400000
;Because the DLL timeouts after 5 seconds, this is done with multiple steps
flash.erase addr=0x004000F0
flash.erase addr=0x0040000F
flash.erase addr=0x004800F0
flash.erase addr=0x0048000F
```

* The BDI does not write implicit any value to the UC3FMCRE registers. If small blocks are used, the appropriate value has to be written to the UC3FMCRE registers via the BDI initialization list or via the connected debugger.

5.2.4 Flash.Load

This command enables loading to flash memory. If the address of a data block is within the given flash range, the BDI automatically uses the appropriate programming algorithm. This command must be executed before downloading is started.

Syntax:

flash.load addr=0x02800000 size=0x200000

addr	The start address of the flash memory
size	The size of the flash memory

5.2.5 Flash.Idle

This command disables loading to flash memory.

Syntax:

flash.idle

5.3 Download to Flash Memory

The BDI supports download and debugging of code that runs out of flash memory. To erase the Flash and to download code add the following entries to the command files PRELOAD.CMD and POST-LOAD.CMD.

Following an example used to download into the flash memory of the MPC860ADS.

PRELOAD.CMD:

```
bdi flash.setup type=am29f size=0x80000 bus=32
bdi flash.erase addr=0x02800000
bdi flash.erase addr=0x02840000
bdi flash.load addr=0x02800000 size=0x200000
```

POSTLOAD.CMD:

```
bdi flash.idle
```

Note:
Some Intel flash chips (e.g. 28F800C3, 28F160C3, 28F320C3) power-up with all blocks in locked state. In order to erase/program those flash chips, use the init list to unlock the appropriate blocks.

```
WM16    0xFFF00000    0x0060        unlock block 0
WM16    0xFFF00000    0x00D0
WM16    0xFFF10000    0x0060        unlock block 1
WM16    0xFFF10000    0x00D0
        ....
WM16    0xFFF00000    0xFFFF        select read mode
```


Supported Flash Memories:
There are currently 3 standard flash algorithm supported. The AMD, Intel and Atmel AT49 algorithm. Almost all currently available flash memories can be programmed with one of this algorithm. The flash type selects the appropriate algorithm and gives additional information about the used flash.

For 8bit only flash: AM29F (MIRROR), I28BX8, AT49

For 8/16 bit flash in 8bit mode: AM29BX8 (MIRRORX8), I28BX8 (STRATAX8), AT49X8

For 8/16 bit flash in 16bit mode: AM29BX16 (MIRRORX16), I28BX16 (STRATAX16), AT49X16

For 16bit only flash: AM29BX16, I28BX16, AT49X16

For 16/32 bit flash in 16bit mode: AM29DX16

For 16/32 bit flash in 32bit mode: AM29DX32

For 32bit only flash: I28BX32

The AMD and AT49 algorithm are almost the same. The only difference is, that the AT49 algorithm does not check for the AMD status bit 5 (Exceeded Timing Limits).
Only the AMD and AT49 algorithm support chip erase. Block erase is only supported with the AT49 algorithm. If the algorithm does not support the selected mode, sector erase is performed. If the chip does not support the selected mode, erasing will fail. The erase command sequence is different only in the 6th write cycle. Depending on the selected mode, the following data is written in this cycle (see also flash data sheets): 0x10 for chip erase, 0x30 for sector erase, 0x50 for block erase.
To speed up programming of Intel Strata Flash and AMD MirrorBit Flash, an additional algorithm is implemented that makes use of the write buffer. This algorithm needs a workspace, otherwise the standard Intel/AMD algorithm is used.

The following table shows some examples:

Flash	x 8	x 16	x 32	Chipsize
Am29F010	AM29F	-	-	0x020000
Am29F800B	AM29BX8	AM29BX16	-	0x100000
Am29DL323C	AM29BX8	AM29BX16	-	0x400000
Am29PDL128G	-	AM29DX16	AM29DX32	0x01000000
Intel 28F032B3	I28BX8	-	-	0x400000
Intel 28F640J3A	STRATAX8	STRATAX16	-	0x800000
Intel 28F320C3	-	I28BX16	-	0x400000
AT49BV040	AT49	-	-	0x080000
AT49BV1614	AT49X8	AT49X16	-	0x200000
M58BW016BT	-	-	I28BX32	0x200000
SST39VF160	-	AT49X16	-	0x200000
Am29LV320M	MIRRORX8	MIRRORX16	-	0x400000

5.4 PPC Interrupt Handling

Almost all PPC interrupts causes an entry into debug mode. By default, the Debug Enable Register (DER) is set as follows:

Debug Enable Register:

Bit	Mnemonic	State	Describtion
0	-		
1	RSTE	enabled	Reset Interrupt
2	CHSTPE	enabled	Check Stop
3	MCIE	enabled	Maschine Check Interrupt
4-5	-		
6	EXTIE		External Interrupts
7	ALIE	enabled	Alignment Interrupt
8	PRIE	enabled	Program Interrupt
9	FPUVIE	enabled	Floating-Point Unavailable Interrupt
10	DECIE		Decrementer Interrupt
11-12	-		
13	SYSIE	enabled	System Call Interrupt
14	TRE	enabled	Trace Interrupt
15-16	-		
17	SEIE	enabled	Software Emulation Interrupt
18	ITLBMSE		Implementation Specific Instructuction TLB Miss
19	ITLBERE		Implementation Specific Instructuction TLB Error
20	DTLBMSE		Implementation Specific Data TLB Miss
21	DTLBERE		Implementation Specific Data TLB Error
22-27	-		
28	LBRKE	enabled	Load/Store Breakpoint Interrupt
29	IBRKE	enabled	Instruction Breakpoint Interrupt
30	EBRKE	enabled	External Breakpoint Interrupt
31	DPIE	enabled	Developement Port Nonmaskable Request

If this is not appropriate for the application the default initialization may be change with an entry in the init list.

```
WSPR    149    0xFFE7400F    ;DER: set debug enable register
```

If the program interrupt is disabled, only hardware breakpoints are supported.
Never disable EBRKE (Bit30) and DPIE (Bit 31) interrupts.

6 Specifications

Operating Voltage Limiting	5 VDC ± 0.25 V
Power Supply Current	typ. 500 mA max. 1000 mA
RS232 Interface: Baud Rates	9’600,19’200, 38’400, 57’600,115’200
Data Bits	8
Parity Bits	none
Stop Bits	1
Network Interface	10 BASE-T
Serial Transfer Rate between BDI and Target	up to 16 Mbit/s
Supported target voltage	1.8 – 5.0 V (3.0 – 5.0 V with Rev. A/B)
Operating Temperature	+ 5 °C ... +60 °C
Storage Temperature	-20 °C ... +65 °C
Relative Humidity (noncondensing)	<90 %rF
Size	190 x 110 x 35 mm
Weight (without cables)	420 g
Host Cable length (RS232)	2.5 m

Specifications subject to change without notice

7 Environmental notice

Disposal of the equipment must be carried out at a designated disposal site.

8 Declaration of Conformity (CE)

CE

DECLARATION OF CONFORMITY

This declaration is valid for following product:

Type of device: BDM/JTAG Interface
Product name: BDI2000

The signing authorities state, that the above mentioned equipment meets the requirements for emission and immunity according to

EMC Directive 89/336/EEC

The evaluation procedure of conformity was assured according to the following standards:

EN 50081-2
EN 50082-2

This declaration of conformity is based on the test report no. QNL-E853-05-8-a of QUINEL, Zug, accredited according to EN 45001.

Manufacturer:

ABATRON AG
Stöckenstrasse 4
CH-6221 Rickenbach

Authority:

Max Vock
Marketing Director

Ruedi Dummermuth
Technical Director

Rickenbach, May 30, 1998

9 Warranty

ABATRON Switzerland warrants the physical diskette, cable, BDI2000 and physical documentation to be free of defects in materials and workmanship for a period of 24 months following the date of purchase when used under normal conditions.

In the event of notification within the warranty period of defects in material or workmanship, ABATRON will replace defective diskette, cable, BDI2000 or documentation. The remedy for breach of this warranty shall be limited to replacement and shall not encompass any other damages, including but not limited loss of profit, special, incidental, consequential, or other similar claims.
ABATRON Switzerland specifically disclaims all other warranties- expressed or implied, including but not limited to implied warranties of merchantability and fitness for particular purposes - with respect to defects in the diskette, cable, BDI2000 and documentation, and the program license granted herein, including without limitation the operation of the program with respect to any particular application, use, or purposes. In no event shall ABATRON be liable for any loss of profit or any other commercial damage, including but not limited to special, incidental, consequential, or other damages.

Failure in handling which leads to defects are not covered under this warranty. The warranty is void under any self-made repair operation except exchanging the fuse.

Appendices

A Troubleshooting

Problem
The firmware can not be loaded.

Possible reasons

- The BDI is not correctly connected with the target system (see chapter 2).
- The power supply of the target system is switched off or not in operating range (4.75 VDC ... 5.25 VDC) --> MODE LED is OFF or RED
- The built in fuse is damaged --> MODE LED is OFF
- The BDI is not correctly connected with the Host (see chapter 2).
- A wrong communication port (Com 1...Com 4) is selected.

Problem
No working with the target system (loading firmware is ok).

Possible reasons

- Wrong pin assignment (BDM/JTAG connector) of the target system (see chapter 2).
- Target system initialization is not correctly --> enter an appropriate target initialization list.
- An incorrect IP address was entered (BDI2000 configuration)
- BDM/JTAG signals from the target system are not correctly (short-circuit, break, ...).
- The target system is damaged.

Problem
Network processes do not function (loading the firmware was successful)

Possible reasons

- The BDI2000 is not connected or not correctly connected to the network (LAN cable or media converter)
- An incorrect IP address was entered (BDI2000 configuration)

B Maintenance

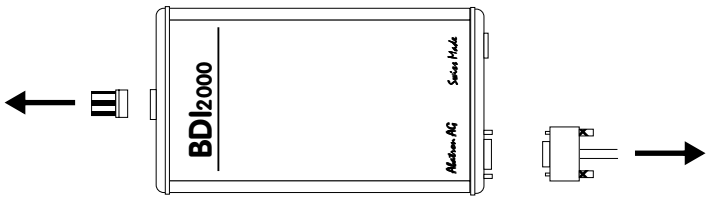
The BDI needs no special maintenance. Clean the housing with a mild detergent only. Solvents such as gasoline may damage it.

If the BDI is connected correctly and it is still not responding, then the built in fuse might be damaged (in cases where the device was used with wrong supply voltage or wrong polarity). To exchange the fuse or to perform special initialization, please proceed according to the following steps:

Observe precautions for handling (Electrostatic sensitive device)
Unplug the cables before opening the cover.
Use exact fuse replacement (Microfuse MSF 1.6 AF).

1

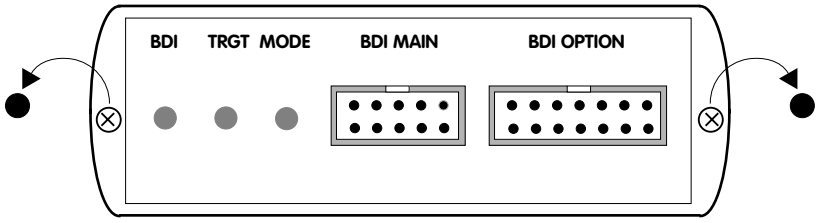
1.1 Unplug the cables



2

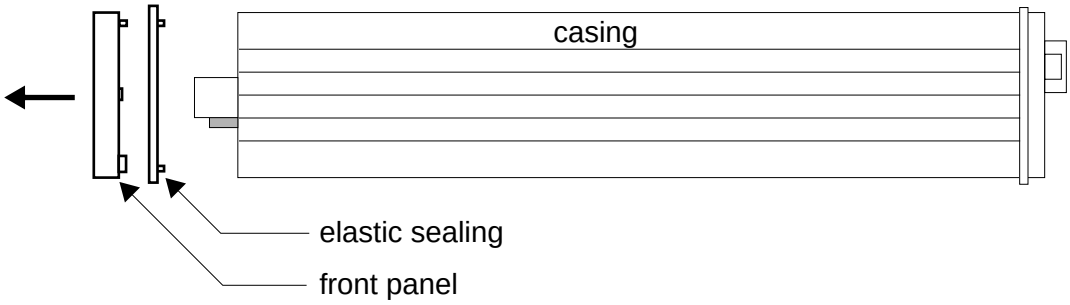
2.1 Remove the two plastic caps that cover the screws on target front side (e.g. with a small knife)

2.2 Remove the two screws that hold the front panel



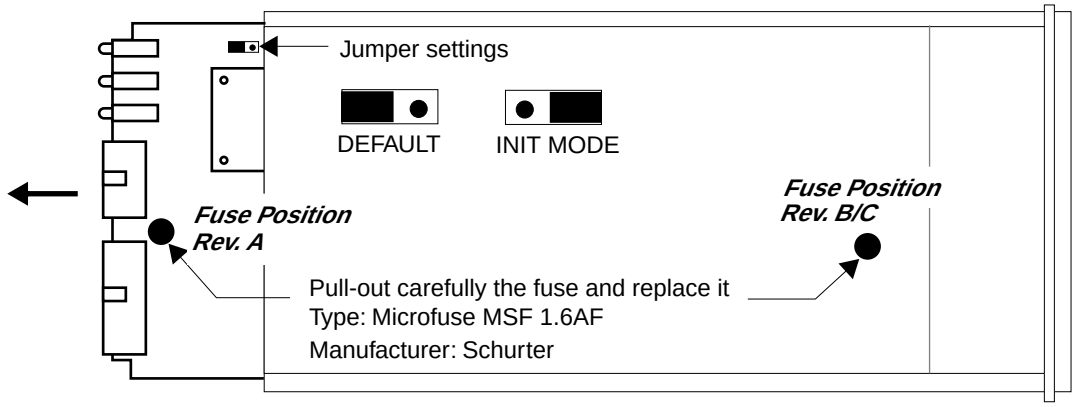
3

3.1 While holding the casing, remove the front panel and the red elastig sealing



4

4.1 While holding the casing, slide carefully the print in position as shown in figure below

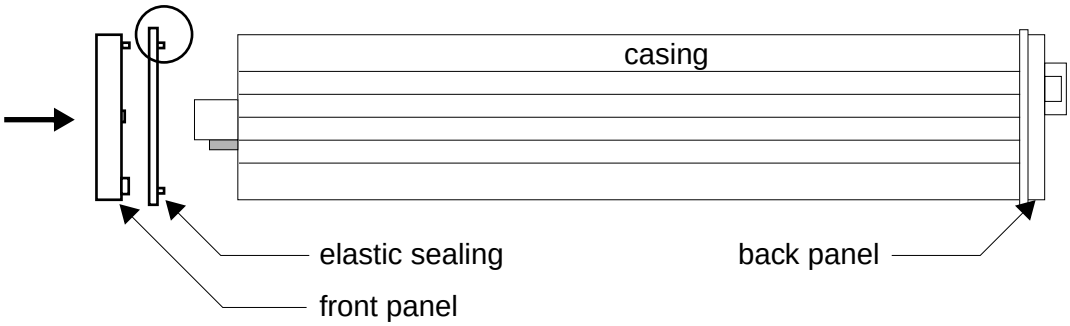


5

Reinstallation

5.1 Slide back carefully the print. Check that the LEDs align with the holes in the back panel.

5.2 Push carefully the front panel and the red elastig sealing on the casing. Check that the LEDs align with the holes in the front panel and that the position of the sealing is as shown in the figure below.



5.3 Mount the screws (do not overtighten it)

5.4 Mount the two plastic caps that cover the screws

5.5 Plug the cables

Observe precautions for handling (Electrostatic sensitive device)
Unplug the cables before opening the cover.
Use exact fuse replacement (Microfuse MSF 1.6 AF).

C Trademarks

All trademarks are property of their respective holders.

